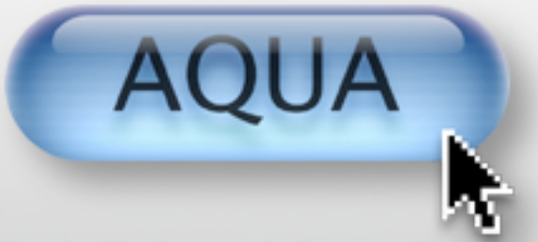
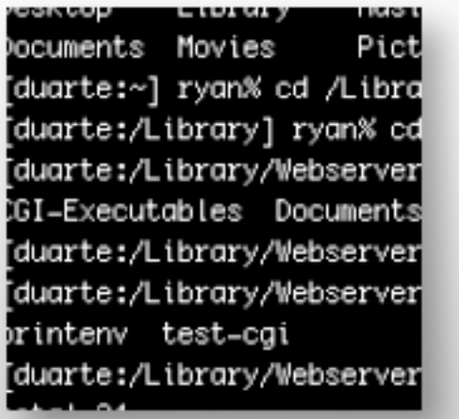
Internet Technologies: Advanced Foundation URL APIs

Session 418





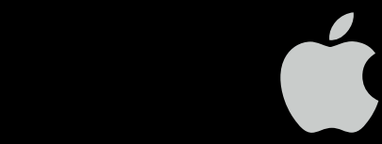
Internet Technologies: Advanced Foundation URL APIs

Ken Kocienda
Safari Engineer
kocienda@apple.com

Introduce new URL loading API

There are new classes and APIs.
But there are also new features.

Disk Caching
HTTP Cookies
Authentication



URL loading?

http://www.apple.com/

ftp://ftp.apple.com/readme.txt

file:///Users/ken/URL-babbling.txt

URL Loading

You have a URL. You want the data.

http://www.apple.com

URL Loading API

Why a new API?

Mac OS X APIs for URL Loading

NSURLHandle in Foundation

NSURL conveniences in Foundation

NSStream in Foundation

CFStream in Core Foundation

CFNetwork in Core Foundation

URL Access Manager in Carbon

Do it yourself

One big reason for developing a new URL loading API…

N
NE
E
SE
S
SW
W
NW

Web browser URL loading is complex

It has to just work

Safari URL Loading Requirements

Support a world-class web browser

Standards compliant

Real-world compliant

Features: caching, cookies, etc.

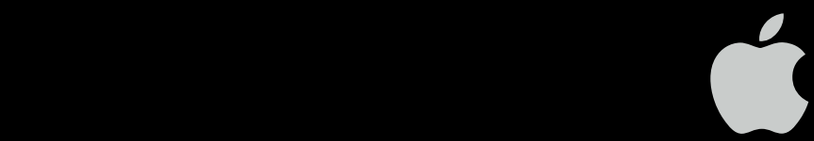
Performance

“If you want to make a fast web browser, you must have fast URL loading.

“If you want to make a fast web browser,
you must have fast URL loading.
Nothing else matters if you
don’t get this right.”

“If you want to make a fast web browser, you must have fast URL loading. Nothing else matters if you don’t get this right.”

Don Melton
Internet Technologies Manager
Apple Computer



While We Were At It…

No Web Kit dependencies

No GUI dependencies

Public API usable outside of Safari

We think this API works great

The Safari URL loading API is now available to you

Who should be interested?

This API Is for You If…

Load URLs

Develop frameworks

Develop applications

Value features more than details

Objective-C or C/C++

Yes, I Did Say C/C++

New API is Objective-C

No technical barrier for C/C++

Here's my advice…

If you like the features…

Link with Foundation and start coding

Example code is available

What changes?

What stays the same?

Safari URL Loading API

Moves into Foundation.
Now called Foundation URL Loading API.

NSURLHandle in Foundation

Superseded, but supported.
Existing code continues to work.

NSURL Conveniences in Foundation

Fully supported.
Internals reimplemented on top of new API.

NSStream in Foundation

Fully supported.

CFStream in Core Foundation

Fully supported.

CFNetwork in Core Foundation

Fully supported.
Foundation URL loading API uses CFNetwork.

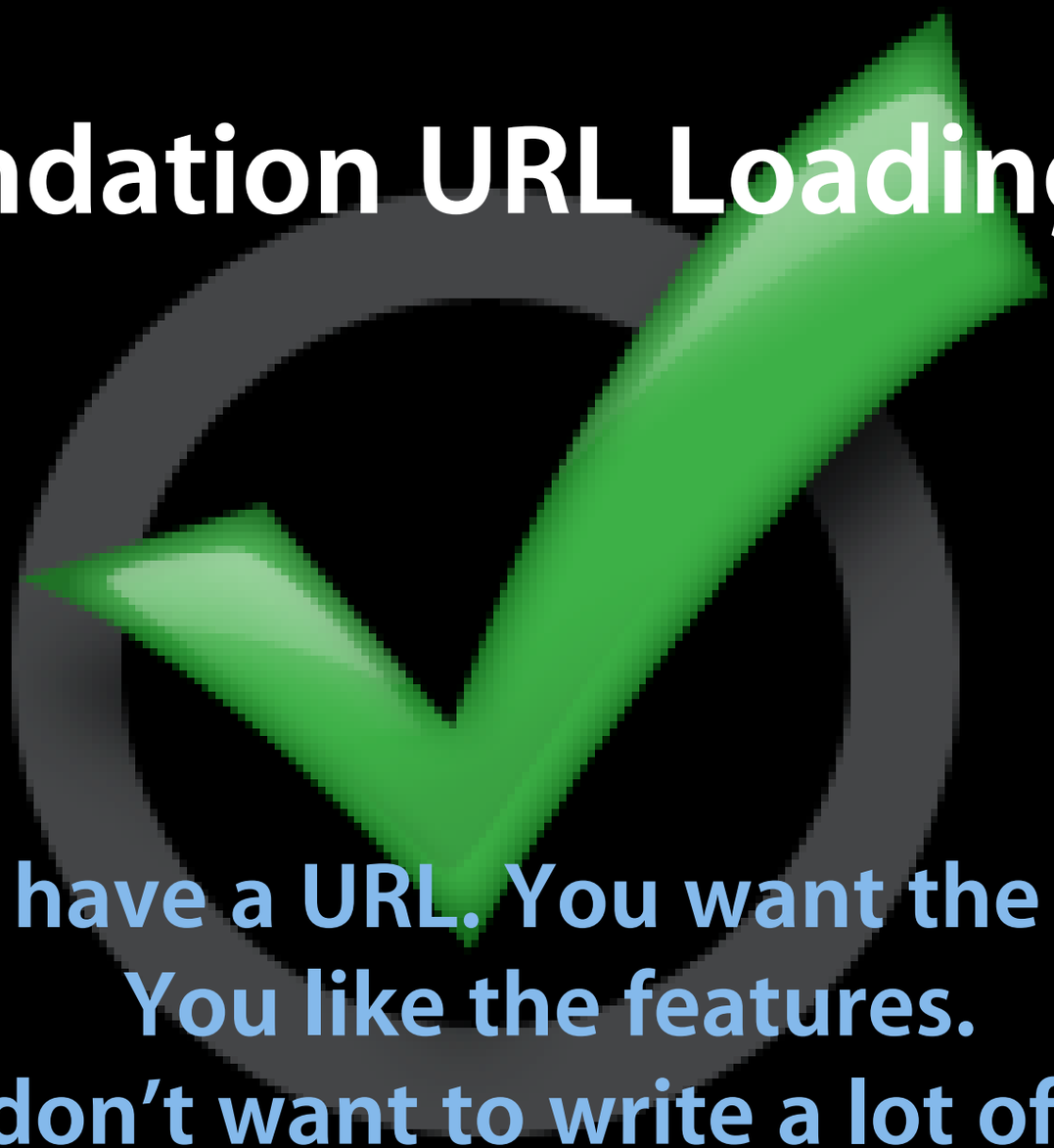
URL Access Manager in Carbon

Still available.
No longer in active development.

You can still do it yourself…

That is a lot of work.

Foundation URL Loading API



You have a URL. You want the data.
You like the features.
You don’t want to write a lot of code.

Foundation URL Loading API

URL Loading

URL File Transfers

URL Content Caching

HTTP Cookies

Authentication

Custom Protocols

Building Blocks

URL Loading

Request

Response

Connection

Delegate Callbacks

URL Loading Building Blocks

NSURLRequest

A Request is a URL plus load-specific data

http://www.apple.com
Bypass cache
User-Agent: MyApp-1.0

URL Loading Building Blocks

NSURLResponse

A Response gives you everything but the data

Mime type: text/html

Content length: 8192

Cache-control: no-cache

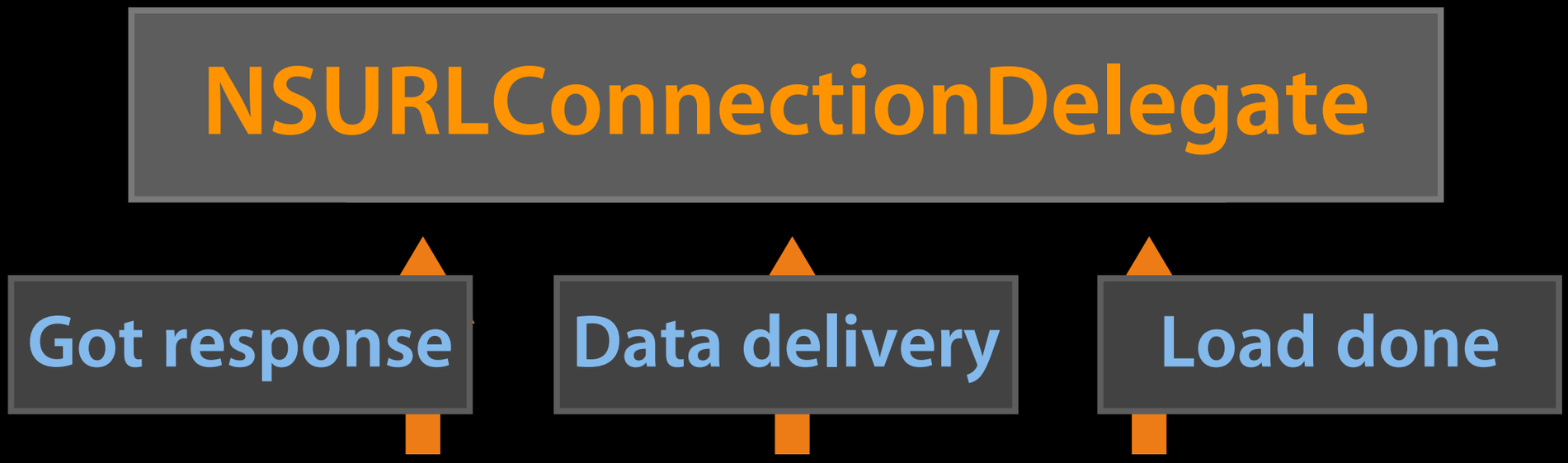
URL Loading Building Blocks

NSURLConnection

A Connection is a handle to an asynchronous load in progress

Created with a request
Uniquely identifies a load
Lets you cancel

URL Loading Building Blocks



Connection

Inform you about events

Chance to take action

Delegate Callbacks

All callbacks are optional

URL Loading Building Blocks

Bringing the pieces together

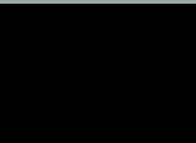
Delegate Callbacks

Request	Connection	Response

URL Loading Code Example

Create a connection and start a load

```
- (void)loadURL:(NSURL *)URL  {

    NSURLRequest *r = [NSURLRequest requestWithURL:URL];

    connection =
        [[NSURLConnection connectionWithRequest:r
                                       delegate:self];

    // Loading starts immediately
}
```



URL Loading Code Example

Implement delegate callbacks

```
- (void)connection:connection didReceiveData:data {
    [allData appendData:data];
}

- (void)connectionDidFinishLoading:con {
    NSLog(@"cool");
}

- (void)connection:connection didFailLoadingWithError:e {
    NSLog(@"uh-oh");
}

@end
```

Demo

URL Loading

Summary

Delegate Callbacks

Request	Connection	Response

Building blocks for URL loading

Asynchronous I/O

Optional delegate callbacks

URL File Transfers

Concepts

URL File Transfers

Very similar to URL loading

Downloads content to disk

A couple extra delegate callbacks

Uploads are coming soon…

URL File Transfers

Building blocks

Delegate Callbacks

Request	NSURLDownload	Response

Download takes the place of Connection

Everything else stays the same

Demo

Foundation URL Loading API

Bringing the pieces together

URL loading and file transfers

Basic building blocks

Remember: delegate callbacks are optional

URL Content Caching

Concepts

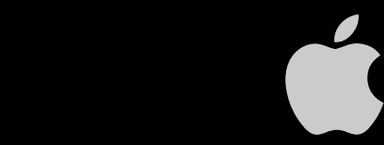
URL Content Caching

Stores loaded URLs locally

Makes future URL requests go faster

Just works…automatically

API provides direct access



Features

URL Content Caching

Good HTTP cache smarts

Disk caching

Configurable policies for request time

Delegate callback for response time

Do You Need this API?

URL Content Caching

Want the cache to work in the background without concern for the details?

You don’t need this API.

You need this API if you want:

Custom cache policies for different loads

Direct access to the cache

Building Blocks

URL Content Caching

Cache

Cached Response

Cache Policies

URL Content Caching Building Blocks

NSURLCache

A Cache is two-level stack:
Memory backed by disk

Works transparently
Look in memory, then on disk
Can be accessed directly

URL Content Caching Building Blocks

NSCachedURLResponse

A Cached Response stores URL data and metadata

Wraps an NSURLResponse
Wraps an NSData
Serialized to disk

URL Content Caching Building Blocks

NSURLRequestCachePolicy

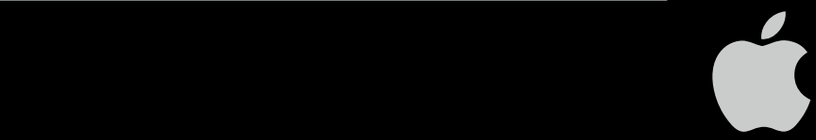
Controls whether cache data is returned for a URL request

- Protocol default
- Bypass cache: reload
- Offline behavior

URL Content Caching Code Example

Access the cache directly

```
NSURLCache *cache = [NSURLCache sharedURLCache];

NSString *s = @”http://www.apple.com”;

NSURL *URL = [NSURL URLWithString:s];

NSURLRequest *r = [NSURLRequest requestWithURL:URL];

NSCachedURLResponse *cachedResponse =
      [cache cachedResponseForRequest:request];

NSData *data = [cachedResponse data];

// do something interesting with data
```



Demo

URL Content Caching

Summary

Cache Policies

Cache

Cached Response

Stores loaded URLs locally

Makes future URL requests go faster

Just works…automatically

Foundation URL Loading API

Bringing the pieces together

Delegate Callbacks

Request

Connection

Download

Response

Cache Policies

Cache

Cached Response

HTTP Cookies

What's a Cookie?

HTTP Cookies

"A packet of information sent by an HTTP server to a World-Wide Web browser and then sent back by the browser each time it accesses that server."

–FOLDOC Free On-Line Dictionary of Computing

Concepts

HTTP Cookies

Enables e-commerce websites

Enables website logins

Just works…automatically

API provides direct access

Features

HTTP Cookies

Persistent cookie storage

Shared per-user among processes

User-configurable accept policies

Real-world compliance

Do You Need this API?

HTTP Cookies

Want cookies to work in the background without concern for the details?

You don’t need this API.

You need this API if you want:

To control cookie accept policy

Direct access to the cookie store

Building Blocks

HTTP Cookies

Cookie

Cookie Storage

Cookie Accept Policies

HTTP Cookie Building Blocks

NSHTTPCookie

A cookie is data received from and sent back to a website

Created automatically
Sent back automatically
Extensive rule checking

HTTP Cookie Building Blocks

NSHTTPCookieStorage

Cookies are stored on disk
and are shared among processes

Manages cookie file
Parses cookie data
API to inspect, set, and remove

HTTP Cookie Building Blocks

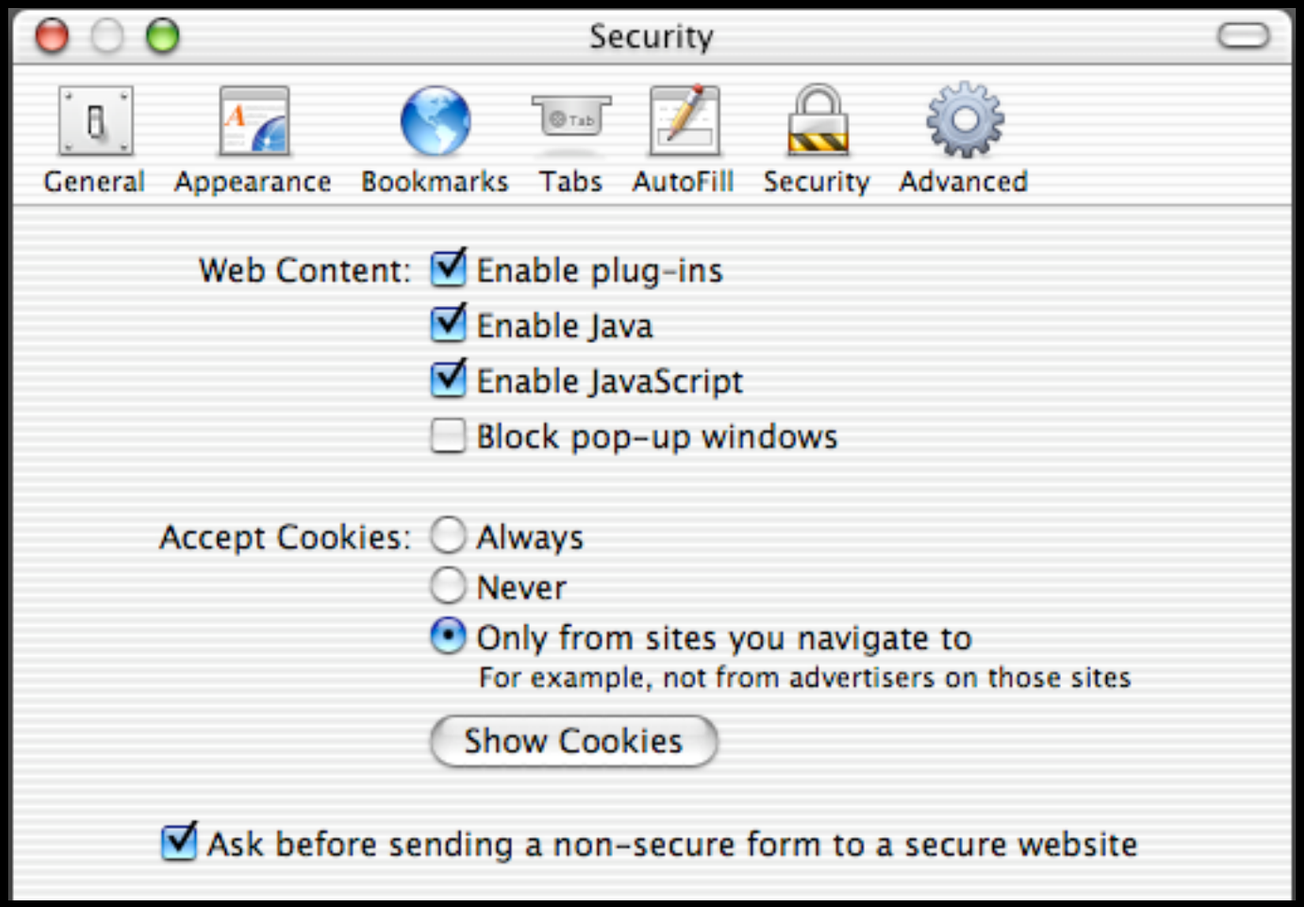
NSHTTPCookieAcceptPolicy

Accepting cookies is governed by this setting

Always accept
Always reject
Accept, but only from same host

Safari Preferences Dialog

Users can choose cookie accept policy



HTTP Cookie Code Example

The code backing the dialog

```
- (IBAction)takeCookiePolicyFrom:(NSMatrix *)radioMatrix {

    NSHTTPCookieStorage *cs = [NSHTTPCookieStorage sharedHTTPCookieStorage];

    if ([[radioMatrix selectedCell] tag] != (int)cookiePolicy) {
        switch ([[radioMatrix selectedCell] tag]) {

        case CookiePolicyAlwaysAccept:
            [cs setCookieAcceptPolicy:NSHTTPCookieAcceptPolicyAlways];
            break;

        // similar case statements for two other policies
```

HTTP Cookies

Summary

Cookie Accept Policies

Cookie

Cookie Storage

Enables e-commerce and website logins

Cookies shared among a user's processes

Just works…automatically

Foundation URL Loading API

Bringing the pieces together

Delegate Callbacks

Request

Connection

Download

Response

Cache Policies

Cache

Cached Response

Cookie Accept Policies

Cookie

Cookie Storage

Authentication

What's Authentication?

Authentication

Proving your identity to a software system.

In this API, authentication usually means logging into a website.

Safari Preferences Dialog

Users can choose cookie accept policy

Features

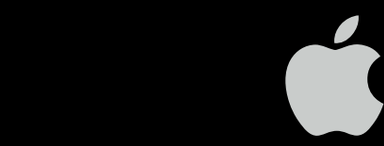
Authentication

Provides hooks for custom UI

No GUI dependencies

Provides for custom auth. schemes

Ask-once capability



Do You Need this API?

Authentication

Loading URLs through Web Kit?
Don’t need to access protected URL data?
You don’t need this API.

You need this API if you want to:
Access password-protected URL data
Develop custom user interfaces

Building Blocks

Authentication

NSURLAuthenticationChallengeSender

NSURLAuthenticationChallenge

NSURLCredential

NSURLCredentialStorage

NSURLProtectionSpace

Demo

Authentication

Summary

Authentication

Provides access to protected data

Ask-once capability

No GUI dependencies

Foundation URL Loading API

Bringing the pieces together

Delegate Callbacks

Authentication

Connection

Request

Response

Download

Cache Policies

Cache

Cached Response

Cookie Accept Policies

Cookie

Cookie Storage

Custom Protocols

Concepts

Custom Protocols

http and https

ftp

file

about

Develop your own

Features

Custom Protocols

No secret APIs

Clean interface to other subsystems

Your protocols are peers

Do You Need this API?

Custom Protocols

Don’t want to write a custom protocol?

You don’t need this API.

You need this API if you want:

To develop a custom protocol

Building Blocks

Custom Protocols

Protocol

Protocol Client

Custom Protocols Building Blocks

NSURLProtocol

Loads URL data in a protocol-specific fashion

Maps to a scheme, like http://

Like a plugin

Must be registered

Custom Protocols Building Blocks

NSURLProtocolClient

Provides the interface to the URL loading system

No secret API

One-way communication

Drives the loading process

Protocol Client

Interface to drive events and callbacks

```
- (void)URLProtocol:(NSURLProtocol *)p didLoadData:(NSData *)data;

- (void)URLProtocolDidFinishLoading:(NSURLProtocol *)p;

- (void)URLProtocol:(NSURLProtocol *)p didFailWithError:(NSError *)error;
```

A Custom Protocol

List of to-do's

```
+ (BOOL)canInitWithRequest:(NSURLRequest *)request;

+ (NSURLRequest *)canonicalRequestForRequest:(NSURLRequest *)request;

+ (BOOL)requestIsCacheEquivalent:(NSURLRequest *)a
                       toRequest:(NSURLRequest *)b;

- (void)startLoading;

- (void)stopLoading;
```

About Protocol
Start loading method

```
-(void)startLoading {
    id<NSURLProtocolClient> client = [self client];
    // create and return response object
    NSURL *URL = [[self request] URL];
    NSURLResponse *response = [[NSURLResponse alloc] initWithURL:URL
       MIMEType:@"text/html" expectedContentLength:0 textEncodingName:nil];
    [client URLProtocol:self didReceiveResponse:response
                    cacheStoragePolicy:NSURLCacheStorageNotAllowed];
    [response release];

    [client URLProtocolDidFinishLoading:self];   // send success
}
```

Custom Protocols

Summary

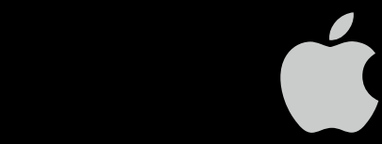
Protocol Client

Protocol

No secret API

Clean interface to other subsystems

Your protocol are peers



Foundation URL Loading API

Bringing the pieces together

Delegate Callbacks

Authentication

Connection

Request

Response

Download

Cache Policies

Cache

Cached Response

Protocol Client

Protocol

Cookie Accept Policies

Cookie

Cookie Storage

Demo

Wrap Up

Roadmap

102 Open Source at Apple Find out how you can get involved	**Mission** **Tues., 10:30am**
411 Safari Technology and Web Standards Taking advantage of standards in Web Kit	**Nob Hill** **Wed., 10:30am**
418 Advanced Foundation URL APIs High-performance URL loading and more	**Nob Hill** **Thurs., 3:30pm**
420 Advanced WebKit APIs Using HTML rendering capabilities in your app	**Mission** **Fri., 9:00am**

Roadmap

426 Internet Technologies: Delivering Web Services Using Web Channels	**Mission** **Fri., 3:30pm**
110 Overview of Mac OS X Networking APIs High level overview of Networking APIs	**Mission** **Thurs.,10:30am**
112 CFNetwork In Depth All the networking knobs and buttons	**Pacific Heights** **Fri., 5:00pm**
FF015 Internet and Networking Technologies	**Mission** **Fri., 10:30am**

Who to Contact

John Geleynse
Manager, Software Evangelism
geleynse@apple.com

http://developer.apple.com/wwdc2003/urls.html

For More Information

Reference Library

Documentation

Web Kit Reference for Objective-C

ADC Home > Documentation > Cocoa >

Web Kit Reference for C

ADC Home > Documentation > Carbon > HIToolbox >

URL Loading System

ADC Home > Documentation > Cocoa > Networking >

Reference Library

Sample Code

WebKit SDK Sample Code

Available in the /Developer/Examples/WebKit folder

Q&A

Internet Technologies: Advanced Foundation URL APIs

http://developer.apple.com/wwdc2003/urls.html

WWDC 2003

Worldwide Developers Conference